A RIGGING CONVENTION FOR ISOSURFACE-BASED CHARACTERS

A Thesis

by

MEGHA NATARAJ DAVALATH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2011

Major Subject: Visualization

A RIGGING CONVENTION FOR ISOSURFACE-BASED CHARACTERS

A Thesis

by

MEGHA NATARAJ DAVALATH

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Tim McLaughlin
Committee Members,	Ergun Akleman
	John Keyser
	Terran Boylan
Head of Department,	Tim McLaughlin

May 2011

Major Subject: Visualization

ABSTRACT

A Rigging Convention for Isosurface-Based Characters. (May 2011)

Megha Nataraj Davalath, B.S., The University of Texas at Austin

Chair of Advisory Committee: Tim McLaughlin

This thesis presents a prototype system for generating animation control systems for isosurface-based characters that blurs the distinction between a skeletal rig and a particle system. Managing articulation and deformation set-up can be challenging for amorphous characters whose surface shape is defined at render time and can only be viewed as an approximation during the process of defining an animation performance. This prototype system utilizes conventional scripted techniques for defining animation control systems integrated with a graphical user interface that provides art directable control over surface contour, shape and silhouette for isosurface-based characters. Once animated, these characters can be rendered using Rendermans RIBlobby implementation and provide visual feedback of fluid motion tests. The prototype system fits naturally within common practices in digital character setup and provides the animator control over isosurface-based characters.

To My Family

ACKNOWLEDGMENTS

First, I would like to thank Tim McLaughlin, who has been an excellent thesis chair, providing me with constant encouragement throughout the course of this thesis. I am grateful to Terran Boylan, from DreamWorks Animation, for rigging the character that inspired this thesis and being an exceptional mentor since the beginning. He guided me through the entire process and answered any questions I had. Huge thanks to Marilyn Friedman and DreamWorks Animation for the cooperation and ability to allow me to work with Terran through this priceless opportunity. I would also like to thank the rest of my committee, Ergun Akleman and John Keyser, for teaching me the key components in understanding the meaning of implicit surfaces and rendering methods.

I am hugely indebted to Jose Guinea Montalvo, a fellow vizzer and good friend, who has helped me from the very beginning in everything from finding the focus of my thesis to explaining concepts about Renderman particular to my implementation.

Special thanks to my Viz companions: Adan Pena, for being my rigging/thesis buddy, Nathan Bajandas, for being the most patient and wonderful animator working with my deadlines, and lastly Julie Pool, for being the creativity behind the character that was used as an example for this system. Other vizzers include David Drell, Bob Graf, Leticia Silva, Tony Piedra, Seth Schwartz and the rest of the VizLab for providing me with an enjoyable memory of my time at Texas A&M University.

Most importantly, I would like to thank my parents, Nataraj and Shylaja Davalath, and my sister, Ranjani Davalath, for the constant love and support they have given me through the ups and downs of life and never letting me lose sight of my dreams.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

The movie Monsters vs. Aliens, by DreamWorks Animation in 2009, defined a need for a hero blobby character that had personality. Visually, B.O.B. was perfect; the shading was beautifully done despite being very difficult to generate UV coordinates on isosurfaces. For being procedurally shaded, the colors and effects added to the surface worked well in every shot. The bubbling effect on the inside also provided a nice touch of detail that blended in with the overall look of the character and added personality. B.O.B. was the main inspiration for this thesis topic . It required solving the problem of combining an unconventional surface with rigging. This character was done for a single purpose for the movie but after realizing there are many other instances such a convention could be used, it was decided that a system that allowed any rigger to produce an amorphous rig would be beneficial.

Due to the development of more powerful and faster computer systems, the use of animated amorphous volumetric shapes has gradually come into use in many sectors of the visualization industry, including: cinema, entertainment, games and information analysis. In movies, characters and effects have required amorphous volumes (smoke monsters, fire entities, water snakes), but most of these were achieved with single-use solutions to be on screen for only a short time. Currently there are no published methods within the framework of standard production practices for defining motion control systems and providing animation control over isosurface shapes. The goal of this research is to provide a convention for the rigger to develop an appropriate motion and control system for the animator on which to animate an isosurface-based character portraying gelatinous or amorphous movement. This is an area of computer graphics currently existing purely in the effects world; therefore there is a need for character driven kinematics.

From a production standpoint, it is important to have robust tools to deliver maximum productivity and output in a minimum amount of time and budget. Therefore, it is necessary to define and develop a convention when something can be repeated across various characters. In the production pipeline of the animation and visual effects industry, the character technical director (character TD) has the role of defining the skeletal and control system for a given character so that it can be given to an animator to give it life [1]. This job entails coming up with a creative and efficient character setup so that it can be versatile and easy to transfer from one character to another [1]. The process of portraying volumetric shapes with believable animation and a convincing performance is a difficult task because current motion and deformation systems do not allow for amorphous movement. Therefore a less complex and inexpensive way to derive a system to portray these types of specialized amorphous characters is needed.

This thesis presents a character motion, control and deformation convention that serves as a prototype for creating isosurface-based, or amorphous, characters. A character can be defined as an embodied representation that moves with articulated limbs and depicts anthropomorphic emotions. The Character Technical Director (TD), also known as a rigger, is in charge of taking a model, laying out the joints, and generating appropriate controls for the motion system that can be passed to an animator. The animator is the artist who sets key poses on the character using key frames to form movement. This system provides the controls that allow the animator to produce animation based on the basic animation principles such as squash and stretch, or inverse and forward kinematics. The system is composed of three principal components:

1. The first involves the generation of a standard joints and control setup based on research of standard production techniques. It is important to have a control system that is designed to an animator's interest. This provides the animator with a familiar

environment, enhancing the value of this system. Due to the amorphous quality of the character, the mesh may not require all appendages to be connected at all times, and will therefore require transient limbs and detachable body parts as a feature of the rig.

2. The second component deals with isosurfaces. Representation of the isosurfaces in the viewport is created using rigid shapes (ellipsoids and other curve primitives) and provides the animator with an approximate silhouette of the isosurface-based character. Renderman's RiBlobby binding class is used to convert this character's surface into high definition renders.

3. The third component of this system is a graphical user interface (GUI) to convert this method into a concise production-ready application. The GUI provides the rigger with a rig creation tool.

This thesis provides a prototype system for generating rigs for isosurface-based characters. Using isosurfaces provides the necessary fluid motion from frame-to-frame compared to other methods. Existing published methods for generating and animating isosurface based characters include those from shape based animation. Animating blend shapes require a large library of shapes necessary to produce simple motion, which can be accomplished with this motion and control system.

This thesis describes a prototype system, provides visual examples of the use of the system, and describes the possible uses and limitations of the system. In the future work section, the trends and needs of professional animation production are discussed relative to this project.

CHAPTER II

PRIOR WORK

Previous research includes examining character design, motion and control systems, isosurfaces, and case studies of professional work involving combination of character animation with isosurfaces. When combined, these areas provide an efficient process through which animators can create the expressive movement of isosurface-based digital characters.

A. Character Design

In order to correctly decide how the model should be constructed and where the joints should be placed, the reference material and artwork are scrutinized in a useful manner. The anticipated range of motion and animated poses of a character give clues regarding the range of motion which helps in designing the control system. The different ways of representing the character for reference was outlined in a 2005 SIGGRAPH course taught by Tim McLaughlin [2].

1. Reference Material

Reference material is one of the most important aspects of understanding and designing a character. There are different ways converting character design and performance requirements into computer graphics techniques. Clues regarding shape and form of the character can be given by analyzing the art work and reference material provided such as:

1. Artwork: The most fundamental type of reference for this genre of work tends to be character drawings or still images depicting a character. Since a lot of animated characters are exaggerated versions of real life creatures or completely make-believe, drawings, as seen in Figure 1, prove to be wonderful references for modeling.

Fig. 1. Left: Artwork and character poses from different angles drawn by Julie Pool. Right: A sample story board drawn by Julie Pool.

2. Reference Images: Having reference photography or drawings is useful to understand the character in terms of size and proportion. Being able to see what your end result should look like makes it easier to work toward a visual goal. In the case of amorphous characters, the study of the nature of liquids would be helpful.

3. Maquette: This is a physical sculpture that is made of clay and then cast in plaster. When constructing the model, the maquette is great reference for information such as size, proportion, density and overall anatomy in comparison to a 2D image.

4. Character Sheet: Character sheets depicting the character in various poses and angles tend to be useful for rigging and articulation for abstract characters (composed of physically plausible elements, but in which those elements have been proportioned or combined is not found in nature [2]).

5. Storyboards: These are fast drawings of places and characters in the scene to demonstrate the story sequentially. Since there are usually characters drawn, especially in action, it is beneficial to use the storyboards to figure out what kind of range of motion and movement will be necessary for articulation.

6. Animatic: They are storyboards, sometimes edited to a soundtrack, and sometimes animated. These are very useful in figuring out articulation and understanding the range of motion for the character joints.

7. Reference Video: Real world reference, or any computer graphics video, is probably the best source of comparison for computer generated work, depending on the level of realism necessary. Video footage shows the realistic movement of what needs to be achieved and can reveal some information regarding articulation, habit and personality of the character.

2. Character Form

Character form typically dictates how a character should be setup. There are crucial elements in understanding the differences of the character form, such as spinal orientation, driving point and locomotion, as it will communicate the issues relevant to setup. The most general forms of a character include:

- Biped: A biped usually refers to a humanoid character that is represented with hips connected to two legs and an orthograde, or upright, spinal system running up to the head region that would be defined as the torso. The arms are typically extended from the upper torso area. The motion for a bipedal character is driven by the hips. Bipeds usually have some of the limbs contacting the ground and other appendages moving freely.

- Quadruped: A quadruped refers to a character that has four legs, typically with the main driving motion being the front two legs. In nature, this type of form can be seen in many animals. Similar to the biped, the quadruped has legs extending from the hip region and a horizontal spinal cord running up near the neck area where the

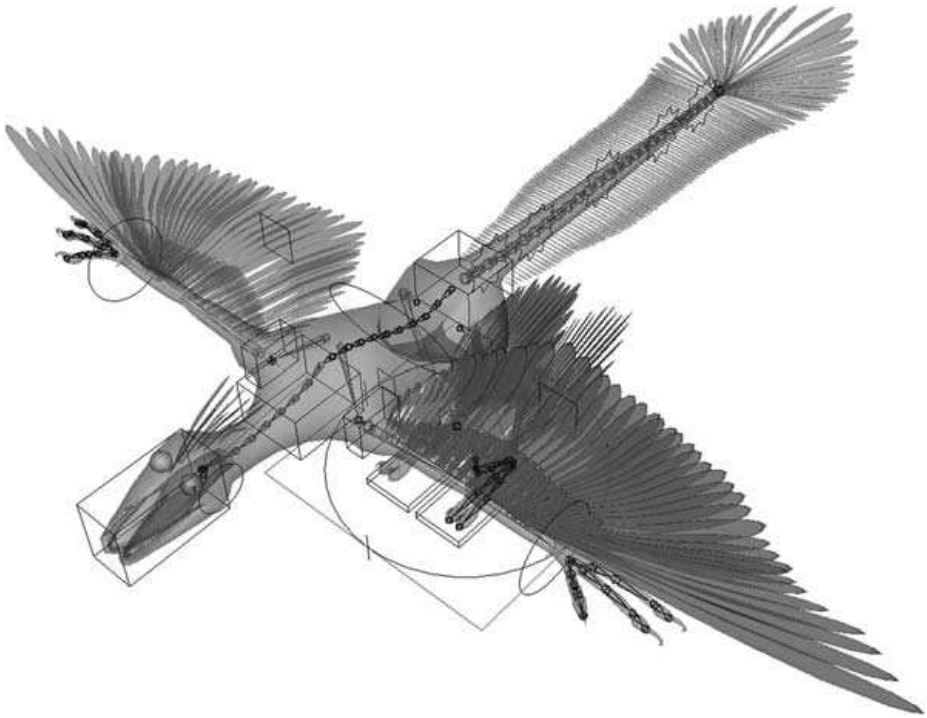

Fig. 2. Winged character rig and model. Character by George Smaragdis and Rob ONeill. [3]

attachments to the front two legs lie[2]. This front connection of the spine drives the appendages. Typically, quadrupeds have all the limbs touching the ground.

- Other Body Types: Most characters can be solved under bipedal and quadruped setups, but there are many other variations that include any number of legs (monopod, tripods or greater than 4), serpentine characters and winged characters [2].

    1. N Legs - Characters that require an unconventional number of legs, such as monopods, tripods or those with greater than four legs, would require careful thought in rig construction. The motion system that would drive this character's movement would determine where the pivotal region of the character lies.
    2. Snake - Snakelike characters are constructed from head to tail as a spinal sys-

tem. They have an undulating motion that travels from the head to the tail in their movement, frequently along a path, and sometimes have legs, like a millipede [2].

3. Winged - The basic form of a winged character can stem from either a biped or quadruped character. It involves an attachment of winged geometry at carefully examined locations on the geometry of a character. The wing system itself is a separate rigging issue in determining how exactly it should act in flight and in rest pose. This form can be seen in Figure 2.

- Facial Features: Facial setup typically consists of eyes and a mouth and visually communicates emotion. Both the eyes and mouth are crucial in expression emotion and the mouth alone is necessary for speaking and lip-syncing, the process of animating the character's face to match the voice or sounds present in the character's voice-over audio track [3]. The facial system is a mixture of deformations and motion setup. Knowing the range of facial expressions and necessity of certain facial musculature movement is helpful when trying to incorporate it into the amorphous character rig. The face is not always an absolute necessity, as an animator can express emotion through specific bodily movements.

3. Motion

The rig is a tool that is used by animators to manipulate the character to provide motion. Understanding and analyzing the motion for the specific character is essential in building the rig. For example, when building a puppet, the range of motion for each appendage is important in attaching the strings to the various limbs. In computer graphics techniques, the expansion of the motion requires more control than puppets. Studying the muscle and joint systems of humans and animals along with their movements and behaviors, the artist

is able to clearly identify the issues involved in creating the rig. The different types of motion that can be studied are explained:

1. Locomotion: Locomotion defines the movement of a character from one position to another. Depending on the body configuration, there are different ways of getting from point A to point B. As previously discussed, there are different types of character setups, and that is crucial in determining the gait. If it is a humanoid biped, then the biped gait would involve two feet that are extended from a hip area and move both legs in a stride, the process of lifting a leg, moving it forward in the air, placing it down and rolling the foot while the other leg starts the similar process. If it is a quadruped, then there will be four legs in motion at different times. The character has to be carefully examined to define what type of locomotion will take place in order to place the joints properly for lower portion of the body.

2. Expression: Emotion is a type of expression that can involve the entire body or through facial changes. Depending on how the character is defined, the expression and personality will determine what is necessary for the rig in order to correctly express emotion [3]. A classic exercise for animators is trying to breathe life into a flour sack rig that has no facial rig. Animating the flour sack is an example of bringing out the basics of animation and trying to express emotions and feelings purely through body language without the need for a face, which tend to be rare. Most characters require a facial setup, and incorporating that into the amorphous rig is a different process than the typical shape based control or clusters setup.

3. Physical Characteristics: For basic setup of the joint system for characters belonging in the same character form category, biped, quadruped or any other, is similar. However, in order to fully satisfy the rig of personal characteristics, special categories should be considered such as weight, center of mass and other independent

attributes. No two characters in this world walk the same way or have the exact same bone structure. Consider the difference in the walk cycles between men and women or adults and toddlers. The shape and proportions of the character can vary from one to another and therefore an older human with long appendages would have a joint setup placed differently than a child. A larger character might have a different center of mass than a smaller version of the same character, and this would influence where the pivotal joint needs to be placed.

B. Motion and Control System

The motion and control system can be defined as the art of designing a digital character system for articulation and a useful control system to convert a models anatomy into a puppet for animation. There are three key components: (a) the motion system, which is a combination of joints that provides the character's skeletal system, (b) the deformation system, which is the process of attaching and modifying the character model based on the movement defined in the motion system and (c) the control system, which is the process of connecting controls to the joints using constraints for specific ranges of motion. [3]. The primary goal of rigging is to create a relationship between the renderable geometry of a model and its underlying joint structure enabling deformation and range of motion, and develop a control system for the purpose of animation.

William Telford, a former student from Texas A&M University's Masters of Science in Visualization Sciences program, studied rigging solutions for the production pipeline for cinema. His thesis research contained a detailed, step-by-step procedure for the rigging process [4]. The need for a simplified control system for a complex rig is very beneficial as it reduces the time it takes the animator to become familiar with the control systems. Telford's research focused on the tools needed to create an effective rig. They include: (a)

forward/inverse kinematics, (b) various transforms (translate, rotation, aim, geometry, analytical and volume), (c) programming and scripting which will be taken into consideration while designing the motion and control system [4]. Providing the animator with a tool that has the ability to satisfy the principles of traditional animation is essential in producing good animation [4].

C. Types of Surfaces

Any type of renderable surface can be used to define a character's surface form. The most common ones are polygonal meshes, subdivision surfaces, NURBS surfaces. Implicit surfaces, though not commonly used.

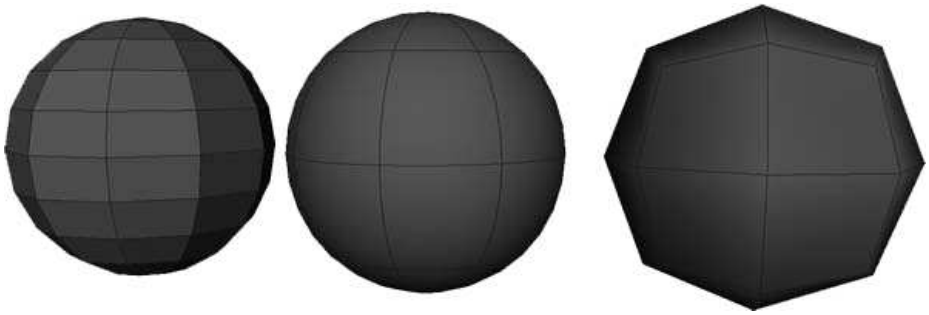

Fig. 3. Surface type sphere primitives (polygon, NURBS, subdivision surface). [3]

1. The polygonal mesh, as seen in Figure 3, is composed of vertices that form polygons, usually 3 sided triangles or 4 sided quadrilaterals, which are then connected to other polygons and form a single three-dimensional polygonal model [3]. Conventional rigging methods use polygonal meshes since there is direct control for weighting the vertices for deformation purposes. This ability for direct manipulation makes it a good surface for many characters, but does not provide the fluid blending effect necessary in portraying amorphous characters.

2. Subdivision surfaces, as seen in Figure 3, represent a smooth surface version of the polygon mesh. It subdivides each polygon in an iterative process and approximates the smooth surface [3]. The base mesh is subdivided iteratively to get finer detail [5]. Since subdivision surfaces are an intricate form of polygon meshes, the rigging process is very similar as clusters are connected to surface vertex members.

3. The Non-Uniform Rational Bézier-Spline (NURBS), as seen in Figure 3, are another geometric type to visualize three-dimensional curves and surfaces. The surfaces are a generalization to Bézier and B-Splines surfaces and have two parametric directions that define the surface and the presence of multi-knots used to represent the curve [5] [3]. NURBs used to be a favored surface type for rigging, but are recently being replaced with subdivision surfaces [3]. The control points would be driven by the joints for control and deformation can be handled with wrappers and weight painting.

Amorphous characters can be achieved with these surfaces with the use of blendshapes that involve modeling every shape particular to animation needs, however this would require a large library of blendshapes and much more time for modeling and setup. Implicit surfaces will now be defined and the complications or ease of integrating such a mesh into rigging.

1. Implicit Surfaces

a. Definition of Implicit Surfaces

The implicit surface, also known as an isosurface, contour surfaces, blobbies, or metaballs, can be represented with a function that define ”the set of points P satisfying the implicit function f(P) = 0” [6]. Metaballs describe an implicit method of modeling surfaces using parametric equations to define points on the surface which can be connected to form a polygonal mesh. An isosurface describes a surface that represents points of a constant

value within a volume of space and is also the result of adding the metaball fields together. Implicit surfaces have the capability of smoothly blending components that are near each other based on a density function.

b. A Brief History of Implicit Surfaces in Computer Graphics

Using real functions to define three-dimensional geometric objects is common in computer graphics. Different functions describe different types of implicit surfaces in both algebra and differential geometry [7]. The inequality, $f(x_1, x_2, ..., x_n) \geq 0$, represents a solid or a volume where as the equation, $f(x_1, x_2, ..., x_n) = 0$ represents the object that is an implicit surface [7].

In 1973, A. Ricci developed a method called constructive geometry which combines simple three-dimensional objects together using operations to form complex objects [8]. Various functions that relate to a particular object can be combined as a single real function that represents a new volume with smooth boundaries by using a sequence of approximating functions [8]. As long as the techniques for computing the contour maps are proper, there should be no difficulties in defining these implicit surfaces [8].

James Blinn took a look at quadric surfaces, including shapes such as spheres, hyperboloids, and cones, as being useful. However, he was restricted in number of shapes and therefore decided to find a better solution that would have an application to a wider range of surface shapes [9]. He wrote the paper wanting to focus on properly displaying molecular models in computer graphics. He noted that for the purposes of animation, the shapes that combine together should have a bonding that contract and stretch based on the distance apart as shown in Figure 4 and through connecting and disconnecting the shapes using exponential operators, the topology of the shapes can also change [9].

Alan Barr, from Rensselaer Polytechnic Institute, added a new collection of parametric objects called superquadric primitives, some shown in Figure 5 [10]. Since most methods

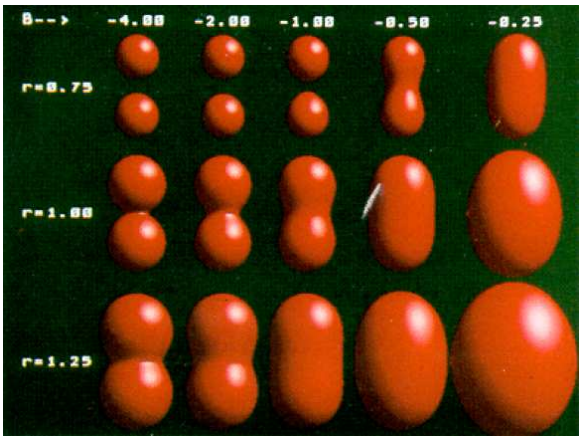

Fig. 4. Object appearance for different blobbiness and radius values. [9]

of representing objects was a list of edges, faces and vertices, Barr developed a method to mathematically define complex shapes and surfaces using a few parameters adding to a existing spectrum of geometric primitives [10].

Jules Bloomenthal developed a set of techniques for creating implicit surfaces, defined as skeletons that are made up of points, splines, polygons or isosurface primitives. Particles can be defined as the center location around which the quadric (spheres or ellipsoids) will be defined. Splines, or curve primitives, are ”a set of central axes for generalized cylinders with possibly varying radii or cross sections” [11]. The last skeleton is the polygon which generates an offset surface based on the defined polygon converted into a set of splines. Manipulations to these skeletons can be done by changing the implicit function, defining it, or changing the weight of the blending function. The offset, referred to as isovalue or contour level, gives control ”in a dilation or contraction of the entire object” [11]. The weighting function defines the blending between the neighboring skeletons and can be additive or subtractive. The center points that define the locations of the metaballs can be directly connected to the joints in the rig for transformation (translation, rotation and

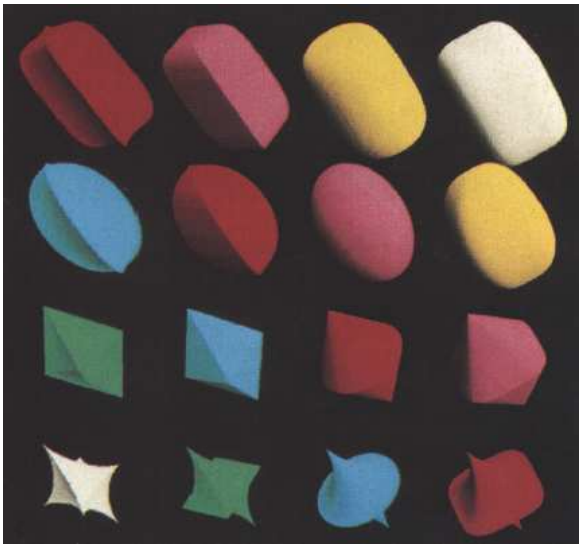

Fig. 5. Superquadric ellipsoids. [9]

scale) manipulation. Current techniques, at that time, provided great difficulty in portraying blended articulated models in a smooth manner [12]. By using three-dimensional convolution method containing polygons and curves, Bloomenthal was able to portray seamlessly integrated convolution surfaces without any bulging [12].

Some of the problems with implicit surfaces are the fact that shape reconstruction is expensive and modeling/shaping implicit surfaces is not as simple as modeling parametric surfaces. Ergun Akleman introduced an efficient method for implicit construction of star solids that are close approximations to the control shapes put together using union and intersections [13]. These computations, using his algorithm, were able to construct star solids in real time that could be molded into human faces [13]. Akleman also contributed to the world of implicit surface modeling through a new formula for the ray-quadrics which is difficult to compute mathematically but simplified the integration into software implementation, creating a shape modeling prototype system conceptually based on ray-quadrics and a method for polygonization for ray-quadric surfaces was also studied [14].

One of the major advantages of the modeling system using implicit surfaces was the automatic blending using the space warping deformations and Boolean operations between primitives [15]. Brian Wyvill describes a method, called *BlobTree* that is able to composite models that use warping, Boolean operations and blending to increase the scope of models defined by implicit surface systems [15]. Using ray tracers and a polygonizer, Wyvill was able to traverse the tree to render the image, a process that would have been very difficult with the systems around at that time [15].

Geoff and Brian Wywill write about the importance of soft objects in the paper ”Data structure for *soft* objects” that could portray clouds, particles, fabrics, mud, water, cushions and living forms [16]. They experimented with the model for soft objects represented by ”a mathematical function defined over a volume of space” and introduced their own data structure and choice of function to control this surface [16]. Soft bodies help in representing natural animation and Wyvill was able to construct surfaces around a set of key control points which were then animated in a physical simulation to produce convincing animation using the modeling technique introduced by Jim Blinn (1982) and Nishimura (1985) [17]. These control points had radius, position, color and other properties, which resulted in a surface that was enveloped around the set of control points and were animation in motion to produce surface movement which was then compared to traditional methods of modeling [17].

The introduction of subdivision surfaces made implicit surfaces less popular in the 1990’s because it was easier to handle extraordinary vertices in modeling. Topological changes, however, in subdivision surfaces was still a problem where as implicit surfaces handle topological changes well. Active implicit surfaces, introduced by Cani-Gascuel and Desbrun in the paper ”Active Implicit Surface for Animation”, are a new method of deformable surfaces [18]. Similarly to Wyvill’s implementation, this paper animates underlying discrete values, or iso-potentials of a field, on a grid rather than the surface directly

[18]. This system presented an optimized implementation that handled surface topological changes during the simulations and generated the metamorphosis between various shapes that didn't have the exact same topology [18].

Witkin and Heckbert presented an approach to controlling and sampling implicit surfaced based on particles [19]. Using constraints to lock a set of particles to the surface, the surfaces were then following the particles in movements [19]. These particles also had control points that allowed for direct manipulation, maintaining the constraint for the surface motion [19]. By using a good sampling distribution of the particles and rendering, they were able to face rapid and extreme deformation changes in the topology of the surface [19].

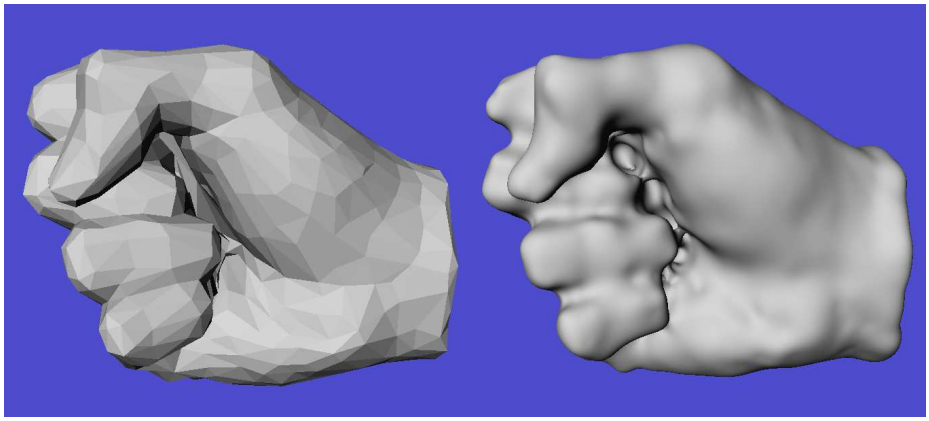

Fig. 6. Polygonal surface of a human fist with 750 vertices (left) and an interpolating implicit surface created from the polygons (right). [20]

A widely used method to convert any polygonal mesh into implicit surfaces using an algebraic solution was developed by O'brien and Turk [20]. Implicit surfaces were used for surface reconstruction, shape transformation, and particularly in this research, model creation. A three-dimensional implicit function is created using locations that specify whether they are interior or exterior of the surface, followed by an interpolation of the data which

define the surface [20]. A major enhancement in this research was the direct manipulation and specification of the location of points and surface normals during model creation since it was difficult to achieve that with existing methods and provided a simple way of converting polygonal models to implicit surface models that blended smoothly as shown in Figure 6 [20].

c. Polygonization of Implicit Surfaces

Most rendering packages require the isosurface to be polygonized. The most common technique for the polygonisation of implicit surfaces is using the marching cubes algorithm, developed by William Lorensen and Harvey Cline [21]. The marching cubes algorithm uses a predefined sized cube. A triangular mesh is drawn in each cube, based on which side of the isosurface its eight vertices are on. Given a single cube, there are $2^8 = 256$ ways to generate a triangular mesh, though similarities between these give only fifteen unique configurations. Once all the cubes have been ”marched” through, the final mesh will be comprised of a set of triangles that can be combined into a single mesh to be rendered. A newer method, called dual marching cubes, presented by Scott Schaefer and Joe Warren, provides a faster and simpler way of extracting the surface information without the subdivision prevalent in the marching cubes algorithm [22]. If there is a change in the implicit surface, calculating these meshes increases the time it takes to render, therefore proving to be a slow process. Methods for rendering the isosurface would be ray tracing and scan conversion, both of which are slow as well. The implementation of the rendering method for isosurfaces in Renderman is unknown, as there is no documentation regarding it. There has been great difficulty with interactive control of implicit surfaces within the viewport of conventional platforms due to the complexity of surface. Animators appreciate interactive visual feedback when they are trying to define a performance, and implicit surfaces are difficult to provide perfect representation in the viewport because of the complexity.

Therefore, the closest method that would provide at least the silhouette of the character would be approximating the character shape with rigid bodies.

D. Case Studies

The first case of amorphous shapes to define a digital character was seen in The Abyss, 1989; the character called the Pseudopod, designed by Industrial Light and Magic in 1989. This character was a series of scans of depth images that was used as keyframes with the in-betweens generated with a morph tool. The alien character in the short film Lifted, by Pixar Animation Studios, was a skinned character rig that had a dynamic fat jiggle deformer to give it the gelatinous motion, so when a character moved quickly and then stopped, the fat kept moving. Other instances that have amorphous volumes are the gelatinous character from Flubber, 1997, the blood creature from Blade, 1998, and the Sandman from Spiderman 3, 2007.

All of the aforementioned characters have been developed in the effects animation department instead of what is traditionally developed in the character setup and animation department. Most of these solutions were achieved using a single-use solution that would have very little screen time. This proves to be highly complex, inefficient and expensive as implementation is pushed to the limits typically in the effects department. Therefore these characters were handled in a manner that required a higher cost outside of the traditional toolset and processes used in character development.

1. Case Study: The Abyss (1989)

Industrial Light and Magic was the visual effects company that worked with James Cameron on the movie, The Abyss, and were required to provide the computer graphics for the pseudopod, the watery snake like character that was an example of a digitally animated creature

Fig. 7. Screenshots from the movie The Abyss showing the Pseudopod. [23]

as seen in Figure 7. The Visual Effects Supervisor, Dennis Muren, was in charge of making sure this computer generated image of an undulating column that was a transparent tentacle was believable to the audience. Computer graphics was not the first choice of action because of the expense and time consumption. The first choice for Muren was to use a clay animation figure, shot with a stop motion technique and have reflections of the water projected onto it, but soon after discussing it with Phil Tippet, a former employee at Lucasfilm who had left to start his own company, they decided to venture into computer graphics as a shell for the clay animation [24]. Cameron admitted It was a leap of faith to use computer graphics. But it was a unique scene. We were trying to create something that had never been seen before [25].

After being asked to create an effect that looked like a giant snake made of water, Murens team developed modeling and animation tests of tubular shapes moving like a snake through the water [24]. Three maquettes of the pseudopod were created using clear resin as a starting point for the computer generated art work. The shape was topologically simple: a tube that had a rounded tip. The pseudopod was given an imaginary spine, that was a line extending lengthwise through the middle and was connected through a series of pivot

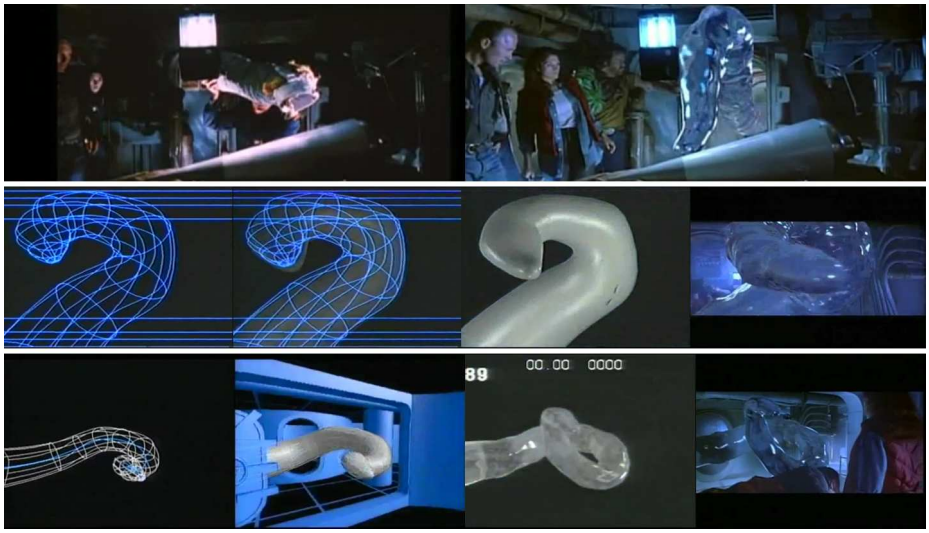

Fig. 8. Screenshots from the Behind the Scenes of The Abyss showing the process of incorporating computer graphics into live footage. [23]

Fig. 9. Screenshots from the Behind the Scenes of The Abyss showing the process of converting facial laser scans into 3D data. [23]

points that behaved like controls that could be manipulated for movement [24]. The biggest task was making it look like an actual column of water that had a random rippling effect and required a more sophisticated technique. This required something beyond just writing a computer program to generate waves, so a program that could control and art direct the shape for a single shot was developed. As seen in Figure 8, the first row shows the usage of a ventilation hose that the crew carried around in place of the Pseudopod character. The second and third rows show the computer graphics technique used by Industrial Light &

Magic to incorporate the character into live footage [24].

Once the basic shape of the creature was modeled and a program was developed to control the motion for each shot, Cameron wanted the tip of the creature to take the appearance of the person it was in front of as seen in Figure 7. Facial renderings, however, was not new to the industry as it had been done in previous movies such as Star Trek IV, 1986, during a dream sequence [24]. However, it was necessary to make a 3-dimensional digitized version of it, so starting with the process that had been done in previous movies, the team decided to grab facial depth scans of both characters in the movie [24]. As shown in Figure 9, these scans were then projected onto the tip of the creature and in-betweens were then calculated to allow it to morph from a rounded tip to the facial rendering with a convincing transformation. These twenty pseudopod shots totaling seventy-five seconds were developed over eight months [24].

2. Case Study: Flubber (1997)

Industrial Light and Magic (ILM) was in charge of the aesthetic, technical and performance issues in this film. They were proposed with the task of finding new solutions to create this shapeless character that is made of sticky goo, solving issues related to light reflection and refraction since it was a transparent character [26]. It was supposed to live and act like a character, everything from portraying emotions, dancing and subdividing itself into several pieces lacking a face and definable body parts [26]. The basic model of flubber could be thought of as blobs of mercury. Animators were struggling to figure out how to animate such a unique character [26]. The rendering of flubber was done through raytracing. It was said to be the best way to represent the transparency, glossiness and show the tiny bubbles that were inside the mesh [26]. Even though it was costly, it also provided the best solution to correctly reflect and refract everything around its environment, including its own appendages. Flubber was the very first movie that used a raytracer for the main

character, which was unique because it was used so extensively since it had to be used in every shot the character was in [26]. Even though flubber seemed like a small addition to the movie, a great deal of work was catered toward this tiny little character that brought life to the movie. This was the first time ILM had done extensive character effects and animation in this manner for a theatrical release [26]. Previously, ILM primarily focused on compositing various animals or dinosaurs into the background plate, which makes color matching and compositing a lot easier [26]. This movie required a completely new type of material and character that you do not see in everyday life, so the integration of flubber in every sequence was a much harder task.

3. Case Study: Monsters vs. Aliens (2009)

Fig. 10. Various screenshots from the movie Monsters vs. Aliens portraying the gelatinous quality of the hero character, B.O.B. [27]

B.O.B., short for Benzoate-Ostylezene-Bicarbonate, was a character in the movie Monsters vs. Aliens, released in 2009 that needed to be amorphous and transform into any shape. Screenshots from this movie can be seen in Figure 10. Since B.O.B. was a hero-character and required significant screen time, it was necessary to provide a different solution which was developing a system to be used in the character and animation setup department. Terran Boylan, a Character Technical Director at DreamWorks Animation, was

the primary rigger who spent ninety-four weeks on the character setup process for B.O.B. [28].

The final character was a blended combination of three versions: (1) a high-resolution NURBS model with no arms, (2) a neutral version with no arms and flattened facial features and (3) an isosurface version with arms [28]. The latter version was composed of triangles combined together to form a mesh and changed topologically every frame. B.O.B had features that included his head coming off, transient arms that would be present or not, bubbles, ground contact, and eyeball detaching [28]. All of these features were designed to be animator friendly as they ”saw a version of B.O.B. with simpler controls allowing for more focus on the performance” and had a control system that resembled a slinky toy [28].

As technology and computer graphics have evolved and reached a point where it is possible to control such implicit surfaces and animate them as hero characters and not effects, it is now necessary to develop a convention.

CHAPTER III

METHODOLOGY

In order to satisfy the art and performance requirements, a system that would enable the rigger to setup the model and rig for any type of character using a modular rig/isosurface setup interface was developed. Being able to develop a rig that would allow animator with creative control for defining the animation performance was key.

A. Animation and Performance Requirements

Animators require interactive viewing of the animation performance in the viewport. This helps in understanding the relationship between poses and arcs necessary in creating compelling motion of a character. Visualizing the isosurface in the viewport in real-time is difficult in any program because the topology of the surface changes every frame and in order to properly generate the mesh in the viewport, the isosurface needs to be polygonized using a meshing algorithm. Depending on the mesh density, generating a new surface for every frame would require high computation and would in turn reduce the playback speed.

In order to work around this, the viewport will contain a close approximation of the silhouette of the final isosurface render using stand in rigid polygon representation primitives. The animator uses a similar interface as conventional rig setups to animate and is able to approximate the final shape using the silhouette of the rigid body stand-ins. It is also easy to view an instantaneous test render of the scene that is an accurate representation of the final high resolution render by setting the quality lower.

B. Tools and Processes

The scripting language used extensively in designing this system was Python, a powerful scripting language that is embedded in Autodesk Maya. The power of scripting methods is the ability to set up a modular and repeatable system that can be incorporated in most common 3D animation packages. Using a common scripting language makes it easily transferable to other animation packages. The modularity of the tool allows the system to be extensible to multiple characters of the same type and similar components of dissimilar characters. The scripts have been broken into compartments for bone placement, building the control system and attaching isosurface functionality to the rig, making it modular while promoting the preservation of current rigging practices.

The Graphical User Interface (GUI) was designed in a way to ease the rigger in building the best possible rig that expects both the artistic and performance requirements. In order to aid the rigger for character creation, the GUI the necessary setup procedures split into different tabs, one for building the rig and another for attaching the isosurface functionality. The modular rigging tab was based on general purpose conventional rigging systems. Everything was broken down into modular compartments of placing locators, joints, and setting up controls. All these processes make it easy to set up any combination of a character and repeat it to multiple characters that tend to be dissimilar. While being able to build a rig of any kind, it also contains the ability to attach isosurface specific features to a rig, such as transient arms. The isosurface tab provides functionality in attaching isosurfaces to the rig. There are options of setting it up with ellipsoids or segment blobs and continuously test rendering various states to make sure the visual aspect matches the artwork.

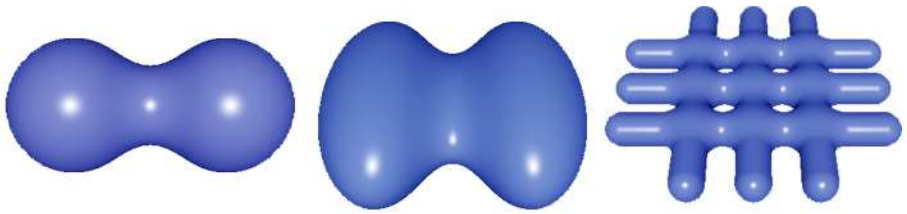

Fig. 11. Isosurface Primitives, left to right: two spheres, two ellipsoids, nine segment chains

C. Isosurfaces

Isosurface describes a volume of space that is the result of adding metaball fields together. In the context of this rig, metaballs will be driven with kinematic joints that will deform the isosurfaces to give the character a smooth blobby look. In addition to controlling single metaballs, segments can be defined as a cylindrical shape that has a start and end point. These have handles attached to certain points on the curve that can be moved and deformed in three-dimensional space. This would provide the animator with greater control of the shape of the blobby mesh.

Renderman's RiBlobby is a surface type that provides rendering for blobby implicit surfaces in Renderman. There are two basic primitive shapes that are included and can be seen in Figure 11, the first being ellipsoids and the second being segment blobs which are rounded cylinders. The ellipsoid starts as a basic sphere and then can be scaled, rotated and translated in any direction to give it the ellipsoidal look. The segment blobs can be thought of as a rounded cylinder that has a start point, end point and a radius.

Isosurfaces have the ability to have different primitives combine together in ways that can appear like they are blending, subtracting from one another, or just intersecting. There are a basic set of operations that allow the primitives to behave in a particular fashion; add and multiply will combine them into a single mesh, subtract and divide will create a

negative body of space and max and min will combine the primitives to intersect with one another. Using these primitives and operations, a character is constructed.

D. Isosurface-Based Characters

There are two basic types of isosurface characters. The first is the process of retrofitting a character with the isosurface functionality and the second is the development of a custom isosurface-based character rig.

1. Converting an Existing Character into an Isosurface-based Character

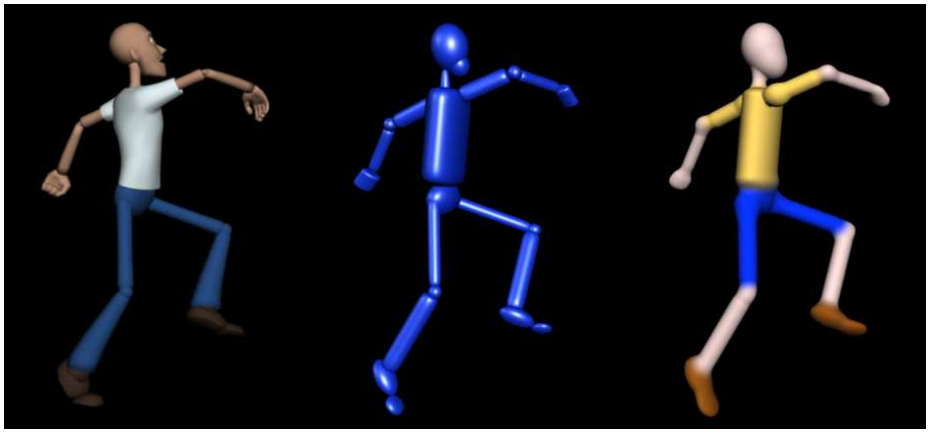

Fig. 12. Left to Right: Original Norman mesh, rigid body stand-in viewport representation, final high-resolution isosurface render

Converting an existing character into an amorphous one is a much simpler task than building one from scratch. Since there is already a set model and potential animation, they can be used as guidelines. The mesh on the rig must be stripped off and templated on a layer for reference purposes. Taking a look at the library of basic primitives for isosurfaces, a decision needs to be made about what shape should be used for which body part. Using the templated model as a reference, the shapes should be modeled into place and attached

to the rig using the various constraints. After setting the render settings to low quality, test renders can show the overall shape of the isosurface to check the progress of the model. Once all of the primitives have been constrained and attached to the rig, various poses can be set and renders will provide results on whether the orientation or complexity of the shapes proved to be accurate or satisfactory to the original model. An example of what the various phases of this setup entails can be seen in Figure 12.

2. Creating an Isosurface-based Character

Isosurface-based characters need special attention because they do not qualify to be a new subset of characters (biped, quadruped, serpentine, etc), but rather a type of surface (polygons, NURBS, etc). Since the amorphous quality visually describes the surface and volume that the character is made of, any character can technically be turned into an amorphous one. Conventionally, by the time a character comes to rigging, the model is finished and ready to be rigged. However, in the case of isosurface-based characters, the model is procedurally constructed during the rigging process. Since blobby characters do not have a well defined range of motion or movement, the rig should be constructed in a way so that appendages and spinal structures bend in every direction that sometime prove unconventional for typical digital characters. The rig should be flexible enough to pose into any formation since it is essentially made up of a gelatinous substance that can take any form.

a. Dissecting a Character

The process of technically executing a character rig from a design or concept requires an analysis of the overall character based on a few guidelines. The guidelines are there to help understand what to look for when trying to develop the technical aspect of a character. The design of the character usually comes from a source such as storyboards, animatics, reference materials or artwork as discussed in prior work. The character sheet proves to be

Fig. 13. Left: Reference material used to develop design of character, Right: Artwork of the orthogonal pose of character by Julie Pool

useful for isosurface-based characters in providing information about the range of volume change that is expected. Storyboards and animatics are key for providing information about form and angles of the character in relation to the camera. Reference images and videos prove to be useful when determining what the final look and movement of the isosurface-based character should be. This is useful in determining which key components of the rig need attention in comparison to others; for example if the sequence of shots that include the character are close ups of just the torso and head, there is no need to rig the legs. The process of looking at the given reference material and dissecting it in terms of looks and motion to understand what would be helpful in building the technical aspects of the character helps getting the character ready to be built properly for articulation. An example of the artwork and poses used in creating the character in this thesis can be seen in Figure 13.

b. Character Form

Any form of character, biped, quadruped, winged, snakelike, etc., can be designed or converted into an isosurface-based character. Because of the surface type and the overall look of the material it is made of, there is a wider range of motion and ability to add special features to the rig in amorphous characters. Transient limbs are specific to blobby characters as they have the ability to appear out of the mass of the body whenever necessary. These appendages have to be built into the rig. There are several ways of attaching them to the body without making it seem obvious with a bulge. The limbs may telescope out of the side with the size of the joint driving the visibility. Visibility can be turned on and they can appear as normal arms would. Depending on the type of character that is being developed, the isosurface nature of the surface will provide the ability to get interesting visual results. For example, blobby snakelike characters should have the ability to curl up into a ball in order to make use of the amorphous ability and combine into a single spherical mass.

c. Motion

After taking a look at the reference images and artwork that are associated with the character, the movement should be analyzed. Movement can be found in references such as character sheets, reference videos, storyboards or animatics as previously mentioned. The gait, personality and individual characteristics should be considered when making decisions on joint placement. Motion can be broken down into four different categories: locomotion, expression, action/interaction and physical characteristics. An example of a character walk cycle and range of motion test can be seen in Figure 14.

1. Locomotion: When rigging the amorphous character, the locomotion plays an important role in determining what kind of motion and control system is necessary for the movement. For a single blobby mass that moves from one point to another

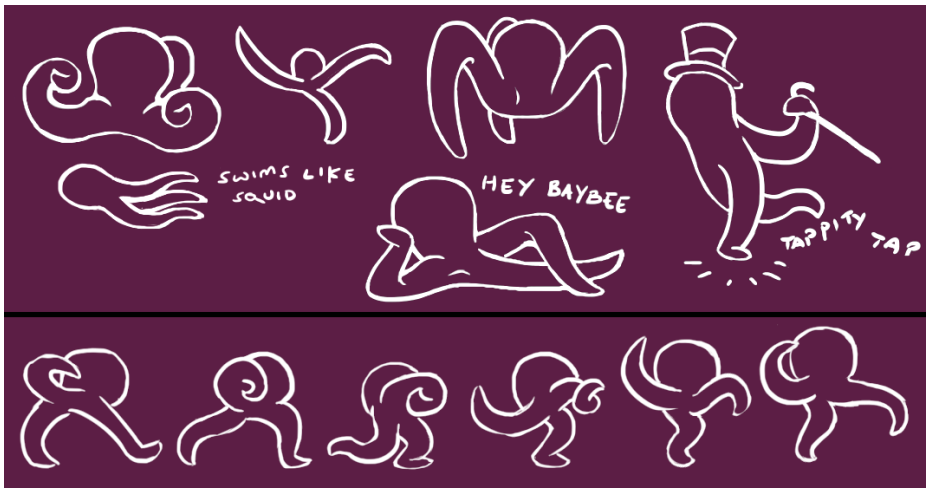

Fig. 14. Top: Character sheet provides insights into range-of-motion for the character, Bottom: Character walk cycle. Drawn by Julie Pool

dragging the body on the floor, there is no need in developing the legs. In order to maintain a correct distance to the ground, the position of the ellipsoids should be carefully placed in the side view right near the ground so there is no penetration into the ground. The terrain and ground that the feet will interact with will behave in a particular fashion different from regular characters. If the character is made of a gelatinous material, then a realistic reaction would be to conform to the ground and shape the bottom to the terrain. In that case, the rigger needs to make sure the environment is set up as a ground repeller so that the isosurface would deform. If the character does not need to conform to the ground, then the rigger needs to make sure to set up the frozen transformations (basically, the rest pose) at a height that is just enough for the bottom rigid isosurface to be standing on the ground.

2. Expression: Expression is essential in performance for any character. Expressive nature encompasses things such as posture, which will give clues to the center of mass or emotion, mostly represented through a system of facial expression. In the case of

an isosurface character, since the rig uses an unconventional mesh, the facial system can be incorporated into the final character rig using negative weights. To carve an eye socket, a negative weighted ellipsoid needs to be placed where the eyeball will sit. To create a cavity for the mouth, a series of isosurface primitives should be positioned together at the specific location. Extra ellipsoids can be used around the cheek area to shape the face. These provide solutions for basic and rough facial features.

3. Interaction/Action: Interaction with other characters and the environment is important to think about while building the rig. Specifically for blobby characters, interaction with the environment is handled differently than other characters. In case the character needs to walk through a table or separate into separate forms in order to interact with a particular object, the rig should be able to handle coming apart and together in a fluid fashion. For blobby characters, interaction with other characters, blobby or not, the rig needs to specifically handle the contact. If there are two blobby characters interacting, it needs to be decided whether they will blend together or not. If they are to be blended, then both the characters need to be under an add operation so they tessellate together as a single mass having a blended intersection. In case they should not blend together, then a max of the two characters should be performed so they are tessellated separately and are allowed to intersect. These decisions need to be taken care of before building the rig, as it will affect how the rig is constructed.

4. Physical Characteristics: After the rig is completed based on the criteria of locomotion, expression and action/interaction, the physical characteristics, or personal attributes specific to the character, such as weight and personality should be looked at before finalizing it.

Based on the methodology instructions discussed, a blobby character should be set up based on the reference material or motivation behind the character. Once a list of guidelines

is clearly defined for the character, the rigger should use the character design sheet to figure out joint placement, range of motion and any other rigging related notes specific for the character. The rigger can build the model based on the given library of isosurface primitives. These isosurfaces can be scaled, translated and rotated anywhere in the scene to properly work together. The character design should be used as a reference in the camera view while building the rough model. The render settings for testing should be set to a low quality so that render time is at a minimum. Once the basic model is satisfactory, the joint system must be built. After templating the primitives in the viewport, the joints should be placed based on the predetermined joint placement locations. Once the rig is constructed, the control system is built based on conventional methods. Other isosurface related controls are also added to the rig based on the character needs. The isosurface primitives are then required to be parent and scale constrained to the appropriate joints based on their range of motion so that they will move and behave like their designated joint. The first draft of the character rig is finished and various poses can be tested and rendered to ensure satisfactory results.

CHAPTER IV

IMPLEMENTATION

A. Reference

To mimic the correct behavior of amorphous shapes, the physical properties of different type of volumes in real life, from gaseous entities all the way to gel materials, were studied. Since masses of bodies are held together in different ways such as surface tension, internal forces or other molecular structures, it was important to study the various options. Since many imitations of real life are not computationally feasible, it was crucial to investigate simplifications to this physical model and isolate which aspects should be analyzed and ignored without sacrificing the desired look. Necessary reference material such as video capture of blob-like substances, movie references and other visual informational resources to study the physics of this kind of movement were gathered in order to reproduce similar visuals specific to movement in computer graphics. Autodesk Maya 2010/2011, Pixar's Renderman Studio 2, Python and QT were used in developing this thesis.

B. Tools

The software that was used to develop the motion and control system was Autodesk Maya 2011. This is an industry standard tool that is used and provides a great interface for modeling and generating a complex motion and control system for characters. It also provides a non-hierarchical method for setting up relationships of objects in the scene from one another using constraints. QT is a cross-platform application framework that is used for building applications and graphical user interfaces. One of the main enhancements from Maya 2010 to 2011 was that the entire interface was developed using QT. Because QT was integrated into this version of Maya, the GUI built for this system was done in QT

as it provided an easy way to create a clean interface. After researching various rendering techniques and softwares, Pixar's Renderman with Autodesk Maya offered sufficient features and functionality to generate a rigging convention for isosurface based character rigs. When the isosurface functionality is added to the rig, the shapes will be generated based on a set of variables such as the primitive shape and the isovalue threshold. The isosurfaces are created using the default primitives, ellipsoids and segment blobs that are in Rendermans RiBlobby implementation. Controls are connected to the rigid bodies in the viewport and the functionality of translation, rotation, scale, and connection to neighboring isosurfaces are available to the animator. The software used for the functionality of isosurface rendering was Autodesk Maya 2010 because that was the version that worked with Renderman.

C. Isosurfaces

Photorealistic Renderman's RiBlobby is a surface type that provides rendering for blobby implicit surfaces using the style of Nishimura et al's *Metaballs*, Blinn's blobby molecules and Wyvill et al's *soft* objects [29]. An important feature of this implementation is the automatic blending that takes place when blobby shapes are placed near each other [29]. Depending on the modeler's choice, the blobby shapes can blend or not blend.

1. Isosurface Primitives

Each primitive in this implementation has an associated opcode that determines the type of shape.

1. Ellipsoids: This primitive, opcode 1001, is the basic sphere that can be scaled, or stretched, in any axis. It is associated with a sixteen floating transformation matrix (4x4 matrix) that determines position, rotation and scale.

2. Segments: This cylindrical primitive, opcode 1002, that resembles a sausage is the other basic primitive that is already included in the binding class. Segment blobs take a twenty-three floating points array that contains three floats for the starting position, three floats for the ending position, a radius and a sixteen floats transformation matrix.

2. Operations

These primitives have the ability to blend together based on a set of operations. After defining the list of primitives using the opcodes listed above, they can be combined together based on their index value with the following operations:

1. Addition/Multiplication: This operation, opcode 0, is the most common as it results in the complete blend of two primitives and renders out blobby, liquid shapes. The fields are added prior to the surface detection, therefore two blobs that are near to each other will result in a large single blob. The binding class requires the opcode followed by the number of primitives taking part in the operation and then the list in the number specified.

2. Subtraction/Division: This operation, opcode 5, takes two values, the first being the primitive and the second value being the primitive that will be subtracted from the first. Either of these fields can be a group of primitives that have previously been defined with an operation. Subtraction is used to poke blobby holes in other blobs and portray negative fields.

3. Maximum: This operation, opcode 2, is used to allow independent fields to cohabitate in the same area. The surface that is rendered out is the result of the largest contributing primitives, and is similar to taking the union of the primitives with no

blended surface. It takes the opcode, followed by the number of primitives for the operation and the index list.

4. Minimum: This operation, opcode 3, can be compared to the intersection function. Followed by the opcode, it also requires the number of primitives and then the index list. The final resulting render will portray the surface that is the intersection of all the shapes being the minimum volume of space shared by all the primitives.

3. RIB Format

A Renderman Interface Bytestream, or RIB format is Renderman's ASCII file format for describing a complete three-dimensional scene into a renderer ready format. Rendermans RiBlobby is a surface type that provides rendering for blobby implicit surfaces in Renderman and explained in Renderman's Application Note #31 [29]. This surface type handles the different isosurface primitives discussed earlier, ellipsoids and segment blobs along with the various isosurface operations, addition, subtract, and per blob specific attributes such as color and overall blobby surface attributes such as threshold or shading type. There are several ways to access blobby information in Autodesk Maya but the most common is a particle system. Since it is necessary to control a single blob to perform the various transformations and attach and move with the animation motion and control system, this system allows better control over the surface definitions. At rendertime, Renderman for Maya grabs the three-dimensional scene and translates it into a RIB format. While the scene is being converted, there is a flag set on one of the objects in the scene to alert Renderman to add a RiBlobby procedure into the RIB file that will contain information regarding how to render the isosurfaces.

The nleaf is an integer that is the number of blob primitives in the scene. The code section is the RiBlobby procedure where opcodes for the primitives will be listed followed

by the start index of its associated transformation matrix. Following the list of primitives, there will be a list of the operations and the associated primitives that is used for the combination. The floats are the list of transformation matrices associated to each primitive. Strings are where the file names for particular z-files relating to repellers would be listed. The parameter list is then an optional list of particular attributes necessary to attach to blob. These can range from specifying color, threshold, or even vertex points for shading.

The RiBlobby procedure format is as follows [29]:

```
Blobby nleaf [ code ] [ floats ] [ strings ] parameterlist
Example:
Blobby 3 [
1001 0 # 0
1001 16 # 1
1001 32 # 2
0 6 0 1 2  # 3
] [
1 0 0 0  0 1 0 0  0 0 1 0   0.89 0   0   1 # 0
1 0 0 0  0 1 0 0  0 0 1 0   0   0.89 0   1 # 1
1 0 0 0  0 1 0 0  0 0 1 0   0   0   0.89 1 # 2
]
```

4. Blobby Attributes

There are attributes that are associated with the blobby scene and four of them have been implemented in this system.

1. Threshold: This global value is computed based on a continuous field function (density value) and is only computed one time per RiBlobby call. The threshold is a value

that determines where the surface should appear when the density is greater than the specified value. It determines how tight or loose the blend for the overall blobby scene will be.

2. Material: Material can be called before the RiBlobby Archive call which means that it can be called one time and will apply itself to the entire scene. The three types of materials that can be specified are Matte, Plastic and Metal. The default has been left to Matte.

3. Colors: The color for each primitive can be specified based on three floating point values that represent Red, Green and Blue. Each representative primitive in the Maya scene has all three attributes that can be set and changed and will then be written into the rib file in the parameter list specified above.

4. Shader Name: This is another option that can be used besides specifying each color. A shader can be created in Renderman SLIM, a shading tool embedded in Pixar's Renderman Studio. The name of the procedural shader can be set to apply the shader to the entire blobby system.

D. Rendering

1. Viewport Representation

The viewport in Autodesk Maya will contain rigid, non-deformable shapes that provide the viewer with a silhouette of the isosurface. Every time an isosurface primitive is added to the scene, it appears in the form and size that will be rendered out. As the shapes are placed near each other, since the default viewport cannot run calculations in approximating the surface mesh in real time, the user will be able to see a silhouette of the overall shape. In order to show negative weights, for operations such as subtraction and division, a negative

cavity will appear in place of the primitive. This was done using boolean operations for the primitives. With a high shading rate and low resolution, test renders are extremely fast in approximating the final render.

2. Generating RIB File

Since the final renders are done using Renderman, the entire Maya scene is first converted to a RIB file format and then sent to the renderer. In order to render out isosurfaces, a special file must be written as a read archive file for the Renderer. A Renderman attribute is attached to a root node for the blobby system and this attribute is flagged during render time by Renderman which evaluates the code. The code is a call to the function that will generate and write out a RIB file specific for the blobby implementation. All the shapes under the root node are traversed, necessary information is obtained and stored as a blobby data structure. The Blobby class in Python has several attributes, a string name, an integer opcode that determines whether it is an ellipsoid or segment, the transformation matrix, vectors for color, position, rotation, scale, start position, end position and a single integer for radius and ID value. The Operator class has fewer attributes; it contains the name, number of children, list of the children by name, id value and type of the operator. As the scene is traversed, three lists are generated, the first one holding every object in the scene including primitives and operators, the second one tokenizing just the isosurface primitives and the third list containing all the operators. After defining the three lists, the primitives list iterates to define a new list instantiating each blobby object based on all the information gathered in the scene. After all the blobby primitives have been instantiated, the list of operators loops through and is instantiated and the children information is set. Once all the objects have been created, the function to write the file is called and all the necessary information is written to a RIB file based on the file format discussed above.

3. Limitations

There were a few limitations that needed to be worked around to solve some issues in this system. The biggest limitation was the ability to match the output of the ellipsoid primitive from the Maya viewport and the Renderman render in terms of translation and scaling. This was tested by having two identically sized polygonal ellipsoids in the scene, one being a stand-in representation for the isosurface shape and the other remaining as a polygon. This issue was resolved after several tests of trying to find the correlation between the two shapes and experimenting by multiplying the isosurface transformation matrix with a particular scale factor. The closest approximation of matching both outputs was solved with a specific scale factor value being multiplied to each value in the transformation matrix of the stand-in polygonal isosurface shape at rendertime. When the isosurfaces are added together the scale provides a close approximation. However, the shapes are subtracted or divided from one another, the workaround scaling issue forces the subtracter to create a negative weight field larger than the stand in representation. In order to fix this, an attribute flag was added to every polygonal primitive in the Maya viewport which would determine if it was a subtractor or divisor and set it to True. If the flag is true, then the transformation matrices relative to those shapes would not be multiplied by the scale factor at rendertime. This provided nicer visual results that closely matched the Maya viewport.

E. Scripting

1. Graphical User Interface

The GUI aids the rigger in generating a rig and attaching the necessary isosurface functionality to the rig in a concise interface. This is also directly connected with Renderman with proper settings so that it can produce the final renders for a given frame interpolating the mesh based on the current state of the rig and associated isosurface-based controls. There

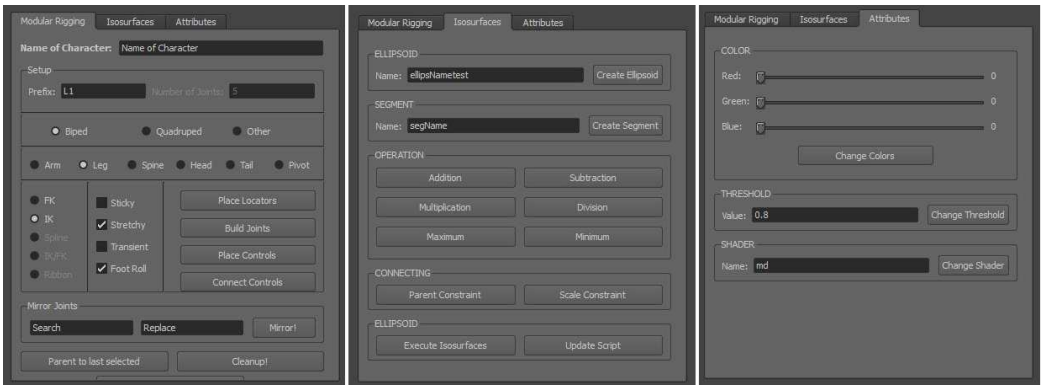

Fig. 15. Left to Right: Modular Rigging tab, Isosurfaces tab, Attributes tab

are three main tabs for the GUI, the first being the Modular Rigging Setup tab, seen in Figure 15, that will be explained shortly. The second tab contains the isosurface functionality, seen in Figure 15. The two basic types of isosurfaces are ellipsoids and segments. You can specify the name of either one and press a button that inserts it into the Maya scene. The next section under this tab consists of the various operations you can perform on the primitive shapes. Another button that is in this tab is the ”Make Blobbies” which adds a Renderman Attribute to the entire blobby system and makes the scene ready for rendering. The third tab is called the attribute tab, also seen in Figure 15, which contains all the value sliders for changing characteristics in the rig. For example, you can change the threshold value, color or shader name.

2. Modular Rigging System

The modular rigging system is a Python scripted system that generates a rig and places joints based on locations of specified locators. The purpose of this system is to compartmentalize the structure of a basic rig in order to build a character of any form or shape made up of legs, arms, spine, head, neck and tail. As seen in the image, the GUI provides

the rigger a concise production ready interface in order to build any character. This system is based on current general character development methods with some features that are isosurface-specific. It is easy to develop a conventional biped or quadruped using this modular rigging system, but can also combine the various features of the system to create unconventional characters. The basic terminology refers to common body parts that are used to create conventional rigs, however, these same body parts can be combined together for amorphous characters for a completely different purpose. For example, the spine system can be used as a long stretchy amorphous mass

The name of the character will define the name for the root of the character once everything is finalized. The first option provided, as a radio button, is the character form, which include biped, quadruped and other. This option mainly determines the direction the joints are built for the spine, as a biped spinal joint chain typically starts at the hip and the quadruped spinal joint chain typically starts at the chest going down to the hip. The next option, also a radio button, includes the type of body part: arm, leg, spine, head/neck, and tail. After selecting the body part, a prefix must be entered. Conventionally this would determine the left side, right side, or center. The reason this prefix is a text box and not predetermined as a radio button is so that there is freedom for the number of appendages or type of character being built.

All the scripts have been broken up into functions that can be used across different appendages, such as a function that builds joints based on a positions array. The number of joints textbox is specifically for the spine, neck and tail option since the number of joints necessary for these appendages varies from one character to another. This option is only enabled when the spine, neck or tail are selected. The first column of radio buttons consist of a group that define the different types of kinematics: forward kinematics (FK), inverse kinematics IK, spline inverse kinematics, a ribbon spine system, and a combination of both IK/FK. These buttons are also enabled based on the selection of body part, the arms and legs

only enable FK, IK and FK/IK switch, spine only enables FK and Spline, and head/neck and tail only enable the option of FK. The next collection of buttons consists of four check boxes and these are attributes specific to the type of body part and kinematic solver. The first two, ”sticky” and ”stretchy”, only work with IK and the spline IK solver. Sticky refers to whether the IK handles move or stay at the current position when the skeleton is being posed by using other IK handles or direct joint manipulation. Stretchy refers to whether the joints will scale in size when the IK handle is pulled past the full length of the joints when placed in a single line. The third is ”transient” and is only enabled when the selection is arm and FK. As previously discussed, the transient arm works like a telescope, so there will be an extra scale control that allows the animator to pull the arm out or hide it. The last checkbox, ”foot roll,” is only enabled in the leg/IK mode, and it adds extra functionality to the leg control, such as twisting the heel or toe, peeling the heel off the ground, standing on the tip of the toe and tapping the toe.

The last column consists of push buttons that actually trigger functions based on the combination of selected choices. The first button, ”Place Locators,” places locators into the scene giving the rigger the ability to move them based on the model. After the locators have been placed correctly, the second button, ”Build Joints” can be pressed to generate the joints based on the position information of the locators. This button also sets up the IK system if it has been selected. The following button, ”Place Controls,” places the controls based on the type of kinematics setup at either the joints locations, IK handles or the clusters for the spline IK. The rigger can then change the translation, rotation or scale of these controllers and then press the last button, ”Connect Controls,” which will then perfectly constrain or parent the correct joints/IK handles to the control. The mirror option takes the selected joint chain and mirrors it across the specified axis replacing the prefix. The last step is placing the Pivot, or the root joint, into the scene and then cleaning up the hierarchy. This modular system allows the rigger to build a basic rig in a click of a few buttons.

3. Languages

Python 2.6.4 is a powerful object-oriented scripting language that is embedded in Autodesk Maya 2011. It is also slowly becoming popular as an industry standard as it is easily embedded in many other softwares, therefore it was decided develop this system being Python-centric as it could potentially be implemented for other softwares. The function calls for the modular system and isosurface generation were scripted using Python. This was useful since it contains wrapped Maya Embedded Language (MEL) calls so it can communicate with Autodesk Maya and also utilize the power of Python. The Graphical User Interface was designed and compiled using Nokia's QT Designer 4.6.3. Once the GUI is designed, QT saves the file as a .UI file, procedurally built python code, which is then called in Maya and easily integrated with the software.

CHAPTER V

CONCLUSIONS

A. Summary

The system developed in this thesis provides a prototype process in developing an articulation, control and deformation system for isosurface-based characters that matches the conventional development process for characters using industry standard tools. The methods provided define instructions for developing a motion and control system for an amorphous character after examining the character design and provided reference material. The concise graphical user interface provides modularity as it compartmentalizes the methods in building a character rig that separates the bone placement, control system and deformation setup. It allows the animator full control over the character for interactive posing and animation in the viewport along with an approximation of the isosurface silhouette using stand-in rigid polygonal representations while being able to test renders using Renderman to generate an accurate image of the isosurface. Several sequences of test animations were rendered out at full resolution and composited beside a rigid-body rendering (as seen in the viewport) for comparison and to provide visual feedback of this unconventional surface rigging system in action.

B. Tests

A series of three basic tests were performed to make sure the various aspects of this system worked.

The first test consisted of a simple four joint FK chain that had a four ellipsoidal primitives stacked on top of each other reducing in size going upwards. This chain was then run through some simple animated poses. The resulting video provided information

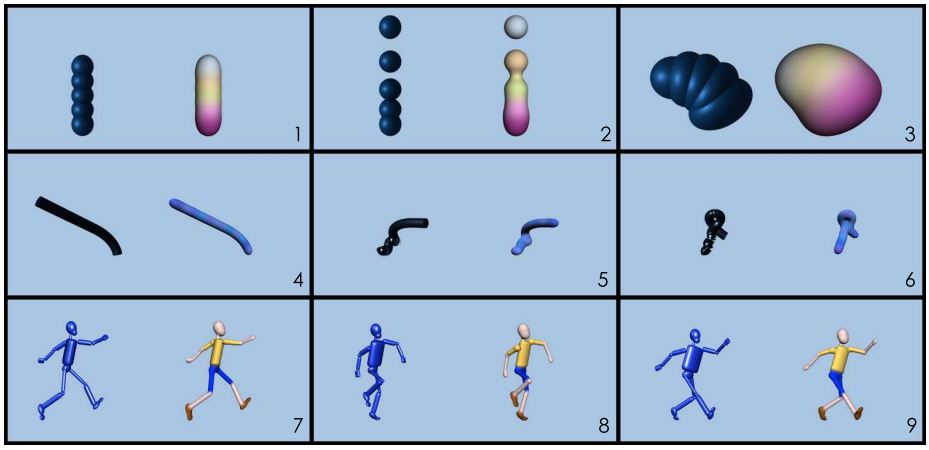

Fig. 16. Top: Ellipsoidal primitives. Middle: Segment Blob Chain. Bottom: Attaching isosurface functionality to the Norman rig

on the fluidity of the motion and the joining and separation of the isosurfaces as it got closer and further away from each other. Sample screenshots can be seen on the top row of Figure 16.

The second test incorporated the segment chain. A long chain of fifty joints, resembling a rope, was created in a single line and a segment primitive was placed for every joint scaled based on the distance between its parent joint and the neighboring joint. This chain was then animated to provide interesting poses with twists and rolled up formations that forced the segments to blend together into a single mass. Screenshots from the resulting video can be seen in the middle row of Figure 16.

The third and final test included attaching the isosurface functionality to preexisting rigs that already had animated walk cycles. The original model was stripped from the rig and templated on a layer to provide a guideline in sculpting the isosurface character. Using the silhouette of the polygonal model, the isosurface primitives were placed appropriately and constrained to the joints in the existing rig. After the rig was complete, the animated sequence of the walk cycle was rendered at a high resolution to provide visual feedback

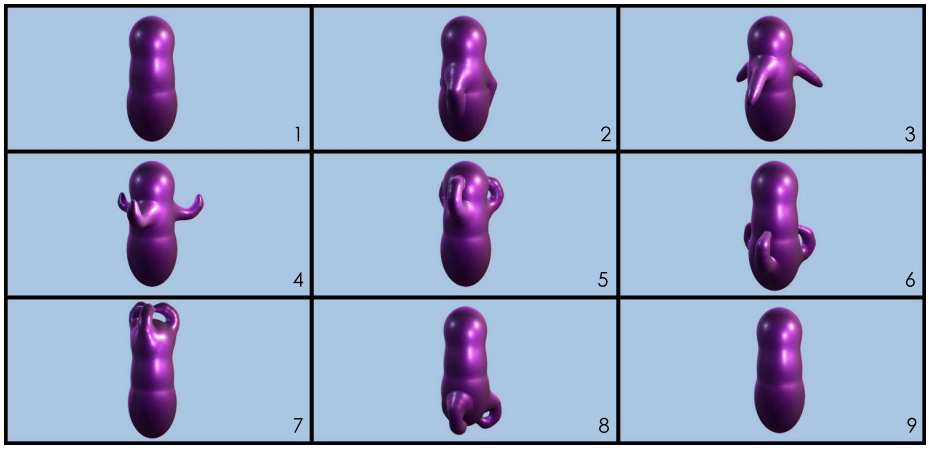

Fig. 17. Screenshots of transient limbs coming in and out of the body in order labeled 1-9

seen on the bottom row of Figure 16 and the supplemental video provided with this thesis.

Figure 17, and the supplemental video, shows an example of transient limbs working with a character. As you can see, image 1 shows the character with no limbs while images 2, 3, 4, and 5 show the limbs protruding from the body and curling up towards the head. Images 6, 7 and 8 shows the limbs moving the original attachment position to the top, middle and bottom of the body. The final image shows the progression of the limbs retracting back into the body from image 8. This is just an example that shows the potential of incorporating transient limbs into any type of character.

C. Isosurface-Based Character Prototype

A fellow student, Julie Pool, from Texas A&M University's Masters of Science in Visualization Sciences program was tasked with designing an isosurface based character that incorporated transient arms. She also provided a character sheet with various animated poses to show the range of motion for the character during the rig generation. The front/side pose of the character design was then examined to figure out where the joints should be placed

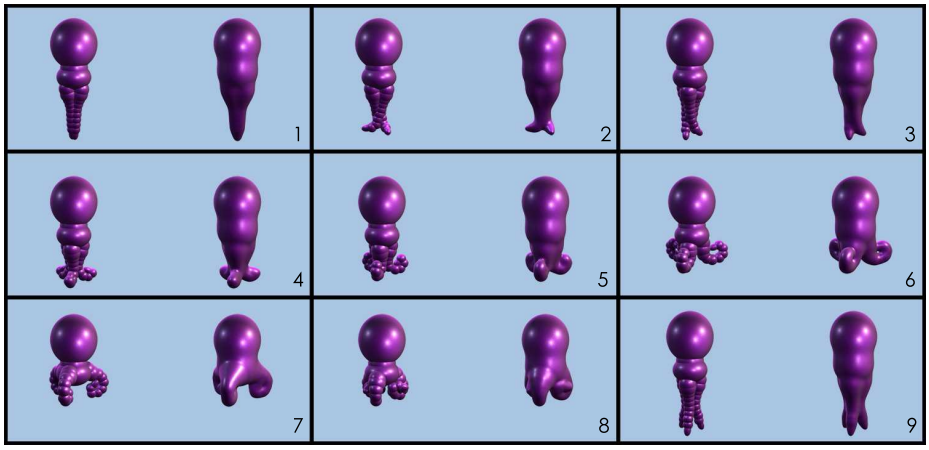

Fig. 18. A few screenshots showing the different callisthenic motions for the blobby character

and what kind of control system was necessary to be able to hit all the poses defined in the character sheet. The image was then brought into Maya as image planes for modeling/rigging purposes. The image plane provided a good way to indicate the locations for joint placements. The GUI was then used to generate the appropriate locators to be placed in line with the joint placement notes on the images. Once the locators were placed, the joint and control system was built. The legs were created using an IK solver and had the ability to stretch when pulled further than the total length of the joints. The spinal system was created using a spline IK solver and the arms were transient with a FK solver. Once the rig was built, the polygonal representation of isosurface shapes were placed into the scene and modeled based on the image planes.

Test renders were done using lower resolution settings to make sure the isosurface final render resembled the shape and look of the character design. Once the character in rest pose was modeled, the rigid primitives were constrained to the joints. The character was then posed to hit some of the character poses provided in the character sheet and test renders were generated to check the similarities and tweak the rig. This character was

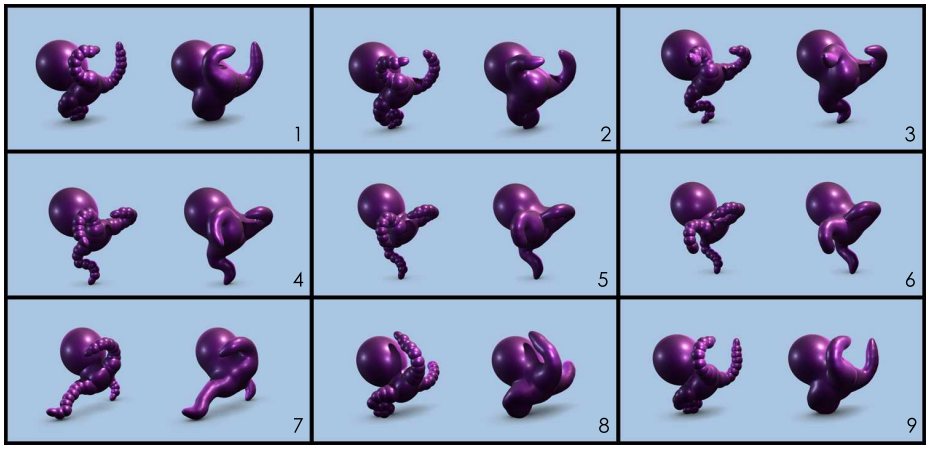

Fig. 19. Screenshots of various phases of the walk cycle

then handed to Nathan Bajandas, another student in the Visualization program, to provide simple animation tests as seen in Figure 18 along with a walk cycle seen in Figure 19. The final video, seen in the supplemental video provided, was then rendered at full resolution.

Test renders were done using lower resolution settings to make sure the isosurface final render resembled the shape and look of the character design. Once the character in rest pose was modeled, the rigid primitives were constrained to the joints. The character was then posed to hit some of the character poses provided in the character sheet and test renders were generated to check the similarities and tweak the rig. This character was then handed to Nathan Bajandas, another student in the Visualization program, to provide simple animation tests as seen in Figure 18 along with a walk cycle seen in Figure 19. The final video, seen in the supplemental video provided, was then rendered at full resolution.

D. Bubbles

Even though the thesis defines the need for amorphous functionality for characters, the system handles attaching isosurface functionality to other forms that are not character based.

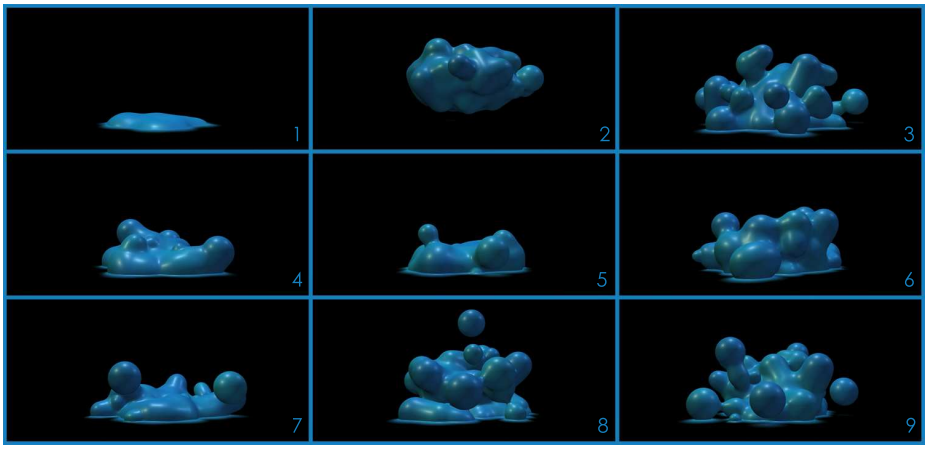

Fig. 20. Screenshots of system working with a particle system

An example of this was a thirty second animated piece that involved a particle system of rigid bodies reacting to music beats. The advantages of using my system instead of the embedded particle system in maya, was that there was direct control over the particles, or isosurface primitive shapes. Using my system, ellipsoidal primitives were created and attached to approximately one hundred particles. The particle position drove the location of the primitives for every frame. The animation of the particle system was generated by grabbing the base information from a piece of music and displacing the translation based on the intensity of the base. After the particles settled at the bottom of the plane, the base translated in the Y direction, forcing the particles to bounce up and down and collide with one another. A large segment plane was placed as the base of the particles so that when they came down, they would blend in with the base to create the appearance of liquid. This example provides a successful way of using the system for non-character based purposes and screenshots can be seen in Figure 20 along with the supplemental video.

CHAPTER VI

FUTURE WORK

This research opens several possible venues in many fields of the visualization industry. In the movie industry this prototype would allow for new directions for animation for effects, art directed physical simulations, and complex character rigging that will match abstract and non-skeleton based concept design and characters.

A. Facial Systems

A pure isosurface based facial system is possible to implement using the current system. Arranging the isosurfaces, shaping the various facial poses using the library of primitives provided, and putting it in the appropriate hierarchy of operations, one can achieve a pretty nice output. Another way to achieve a good facial system, one way to have full control is to blend a conventional surface type that provides facial features with the isosurface, similar to the approach taken with constructing B.O.B. from Monsters Vs. Aliens. Generic convention for facial animation uses a NURBS or polygonal based surface mesh with a combination of direct joint manipulation and blendshapes that give direct sculpting, or modeling, access to the various poses from a facial shape library. Other methods include clusters, which is also similar to the direct surface manipulation to generate the various facial poses. For this system, the facial system can be developed in one of the conventional surface types and later warped with the isosurface based model. This would have to be taken care of at render time and since it is the process of combining various meshes together, it requires research for implementation.

B. Motion Blur

Motion blur is defined as the in-between frame calculations. When an object is in motion, there tends to be a streaking effect in long exposure when trying to capture a single image. Therefore, it is only natural for this blurred or streaking effect to appear when rendering an animated sequence in 3D. In order to do so, the current render frame must know information about the previous and next frame in the sequence of motion in order to calculate and apply the necessary effect. Since the current topology of surface meshes are the same for every frame, it is easy to calculate a given point from one frame to the displacement in the next frame. Isosurfaces, however, change topology per frame, so it is a completely different avenue of research to calculate motion blur between frames as Renderman currently does not handle motion blur for blobbies. Implementation may entail incorporation of per vertex attributes to find the closest point to the surface for each frame and calculate motion blur as a post process. Also, the velocity vectors of each primitive can be calculated into the isoblending so the result will be a linear blend of velocity vectors.

C. Viewport

Many areas of future work exist from the viewport all the way to final renders. In the current setup, the viewport contains rigid bodies that provide the animator with the basic silhouette of the character. Until the animator renders a frame, an isosurface mesh will not be visible. In order to help the animator see what the motion looks like in the viewport, a plug-in could written to tessellate meshes using a meshing algorithm for mesh consistency that matches the output of Renderman's results based on the metaballs information provided in the rig in real time. This would require knowledge of Renderman's internal algorithms. This would be a valuable asset for the animator, as it would decrease the amount of time used for testing renders to make sure the poses are perfect and be able to see the shapes in the viewport.

D. Ray Tracing

A ray tracer could be written to directly generate isosurface renders. This could be written as a plugin for Maya or a standalone renderer. The information of the location, scale and rotation of the metaballs from the Maya scene could be fed into this ray tracer that would then render isosurfaces. The advantage of this would be specialization and fine tuning the renders based on the user interest. The majority of the work in the implementation of this thesis was writing out RIB files in the format Renderman would understand for rendering. It was limited to what Renderman was capable of. However, if there are characteristics that cannot be achieved in Renderman, a custom ray tracer that was treated like open source software and personalized based on need, would prove to be useful.

E. Density Function

Isosurfaces are generated based on the location and density value of the metaballs. One area of work could be incorporating control over the density functions to provide various and better boundaries between the metaballs into the current system. Control over different density functions for the connection of specific primitives within the same system would also result in interesting images. This would have to be done in a different renderer as it involves writing code to define various functions on where the surface should appear. Currently, the implementation used in Renderman's RiBlobby binding class uses the basic density function that would produce a sphere. Other functions that could be incorporated would provide interesting results in surface shape. This could also be added as a feature to a raytracer that renders isosurfaces.

F. Effects

Since isosurface-based character rigging blurs the boundaries between character rigging and character effects, there is a whole world of opportunities to incorporate effects into the rig. Everything from the bubbling reaction to the ground all the way to internal effects because of the transparent quality of the surface would fall under character effects. Being able to handle interactions with other objects that require the surface mesh to deform would also require an effects system added on to the rig. A mass spring system could be added to the main particles in the spine so that during movement, the particles would wobble. Adding a feature of density or viscosity to the model would be an interesting aspect the animator could have control over. The blobby character does not need to be made of a transparent liquid material, but rather a whole class of other physical materials such as fire, magma, clouds, smoke or mercury. These types of physical material attributes need special attention while building the rig and also require a layer of effects to be added to the mass already existing with the isosurface.

REFERENCES

[1] M. Ford and A. Lehman, *Inspired 3d Character Setup*, K. Clark and M. Ford, Eds. Premier Press, Inc., 2002.

[2] T. McLaughlin, "Taxonomy of Digital Creatures: Interpreting Character Designs as Computer Graphics Techniques," in *ACM SIGGRAPH 2005 Courses*, ser. SIGGRAPH '05. New York: ACM, 2005.

[3] R. O'Neill, *Digital Character Development: Theory and Practice*, ser. The Morgan Kaufmann Series in Interactive 3D Technology. San Francisco, CA, USA: Morgan-Kaufmann (Elsevier), 2008.

[4] W. L. Telford, "Rigging skeletal perissodactyl and artiodactyl ungulate limbs using analytic inverse kinematic-based solutions for a feature film production environment," Master's thesis, Texas A&M University, 2007.

[5] Autodesk, "Autodesk Maya Online Help," 2010. [Online]. Available: http://download.autodesk.com/us/maya/2011help/index.html

[6] J. Bloomenthal, "Polygonization of Implicit Surfaces," *Comput. Aided Geom. Des.*, vol. 5, pp. 341–355, November 1988.

[7] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko, "Function Representation in Geometric Modeling: Concepts, Implementation and Applications," *The Visual Computer*, vol. 11, pp. 429–446, 1995.

[8] A. Ricci, "A Constructive Geometry for Computer Graphics," *The Computer Journal*, vol. 16, pp. 157–160, 1973.

[9] J. F. Blinn, “A Generalization of Algebraic Surface Drawing,” *ACM Trans. Graph.*, vol. 1, pp. 235–256, July 1982.

[10] A. Barr, “Superquadrics and Angle-Preserving Transformations,” *IEEE Computer Graphics and Applications*, vol. 1, pp. 11–23, 1981.

[11] *I3D ’90: Proceedings of the 1990 Symposium on Interactive 3D Graphics*. New York: ACM, 1990.

[12] J. Bloomenthal and K. Shoemake, “Convolution Surfaces,” in *SIGGRAPH*, 1991, pp. 251–256.

[13] E. Akleman, “Interactive Construction of Smoothly Blended Star Solids,” in *Proceedings of the Conference on Graphics Interface ’96*. Toronto: Canadian Information Processing Society, 1996, pp. 159–167.

[14] ——, “Interactive Construction of Ray-Quadrics,” in *Proceedings of Implicit Surfaces ’98*, 1998, pp. 105–114.

[15] B. Wyvill, A. Guy, and E. Galin, “Extending the CSG Tree - Warping, Blending and Boolean Operations in an implicit surface modeling system,” *Comput. Graph. Forum*, vol. 18, no. 2, pp. 149–158, 1999.

[16] G. W. C. McPheeters and B. Wyvill, “Data Structures for Soft Objects,” *The Visual Computer*, vol. 2, pp. 227–234, 1986.

[17] B. Wyvill, C. McPheeters, and G. Wyvill, “Animating Soft Objects,” *The Visual Computer*, vol. 2, no. 4, pp. 235–242, 1986.

[18] M. Desbrun and M. Cani-Gascuel, “Active Implicit Surface for animation,” in *Proc. Graphics Interface ’98*, 1998, pp. 143–150.

[19] A. P. Witkin and P. S. Heckbert, “Using particles to sample and control implicit surfaces,” in *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’94. New York, NY, USA: ACM, 1994, pp. 269–277.

[20] G. Turk and J. F. O’brien, “Modelling with implicit surfaces that interpolate,” *ACM Trans. Graph.*, vol. 21, pp. 855–873, October 2002.

[21] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3d surface construction algorithm,” in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’87. New York: ACM, 1987, pp. 163–169.

[22] S. Schaefer and J. Warren, “Dual marching cubes: Primal contouring of dual grids,” *Computer Graphics Forum*, vol. 24, no. 2, pp. 195–201, 2005.

[23] J. Cameron, *The Abyss*. Twentieth Century Fox Film Corporation, 1989.

[24] D. Shay, *Dancing on the Edge of the Abyss*, ser. Cinefx, 1989.

[25] D. Muren, “The Abyss: Pseudopod Sequence,” *Prix ARS Electronica*, 1990. [Online]. Available: http://90.146.8.18/en/archives/prix_archive/prix_projekt.asp?iProjectID=10789

[26] T. Bertino, “Flubber,” *Prix ARS Electronica*, 1999. [Online]. Available: http://90.146.8.18/en/archives/prix_archive/prix_projekt.asp?iProjectID=12730

[27] C. Vernon and R. Letterman, *Monsters Vs. Aliens*. Paramount Pictures, 2009.

[28] R. Dunlop, “Cgsociety - Monsters vs Aliens,” March 2009. [Online]. Available: http://features.cgsociety.org/story_custom.php?story_id=4985

[29] Pixar, “Application note 31: Blobby implicit surfaces,” September 1999.

VITA

Megha Nataraj Davalath

Visualization Laboratory

Texas A&M University

C108 Langford Center

3137 TAMU

College Station, TX 77840-3137

c/o Tim McLaughlin

megha@viz.tamu.edu

Education

M.S., Visualization Sciences, Texas A&M University, 2011

B.S., Computer Science, The University of Texas at Austin, 2008

The typist for this thesis was Megha Davalath.